Processus de branchement des réseaux de Petri à reset arcs

Maurice Comlan[1], David Delfieu[1] and Médésu Sogbohossou[2]

[1] Institut de Recherche en Communications et Cybernétique de Nantes, France
{maurice.comlan, david.delfieu} @irccyn.ec-nantes.fr
[2] Université d'Abomey-Calavi, Bénin
medesu.sogbohossou@gmail.com

Résumé

Les réseaux de Petri sont un outil très répandu de modélisation et d'étude des systèmes concurrents autour desquels s'est constituée une large communauté scientifique. Il existe des extensions qui ont pour but d'étendre le formalisme de référence par les réseaux de Petri pour proposer soit des modèles plus compacts en termes de description, soit plus puissants en termes de pouvoir d'expression. Nous pouvons citer entre autre l'ajout des reset arcs qui ajoutent un pouvoir expressif utile en ce sens qu'ils permettent de remettre à zéro (notion de réinitialisation) le contenu d'une place sans impacter sur les règles de sensibilisation des transitions dans le réseau. Nous proposons pour les réseaux de Petri à reset arcs ayant un réseau normal sous-jacent borné, une approche pour construire le dépliage (méthode d'ordre partiel pour contourner le problème d'explosion combinatoire lié à la construction du graphe d'états) et d'un préfixe complet et fini du dépliage. Nous présenterons aussi un algorithme pour l'identification des processus de branchement issus d'un tel dépliage.

1 Introduction

Les systèmes sont de plus en plus très complexes (multitâches, réactifs, ...), s'exécutent sur des architectures mono ou multiprocesseurs, parallèles ou répartis. Les tâches peuvent être soumises à des contraintes de synchronisation, de concurrence, de communication, etc. De par leur complexité, de tels systèmes sont difficiles à appréhender et il est nécessaire de disposer d'outils et de modèle fiables pour, d'une part, les concevoir et les comprendre, et d'autre part, pour les vérifier en s'assurant qu'un certain nombre de contraintes sont bien respectées. Par exemple, on peut s'assurer de l'absence de blocage, garantir le non dépassement de capacité d'un buffer à capacité limitée, garantir l'exclusion mutuelle sur les ressources, prouver que les contraintes temporelles sont bien respectés et bien d'autres. La finalité est de garantir la fiabilité et la sureté du fonctionnement du système et pour cela, il faut, pendant toute la phase de conception, de disposer d'une approche permettant d'expliciter précisément le comportement du système et d'analyser ce comportement afin de s'assurer qu'il est bien conforme au cahier de charges du système [12].

Pour une approche fiable, on a recours à des méthodes formelles qui permettent une modélisation mathématique du fonctionnement du système à partir de laquelle il est possible d'effectuer des preuves. Il s'agit en fait d'une représentation abstraite et approchée du système réel. Plusieurs modèles existent et chaque modèle possède ses caractéristiques propres, plus ou moins pertinentes dans une conception spécifique. En conséquence, le choix du modèle dépendra du système conçu et des propriétés à analyser. Parmi l'ensemble des modèles existants, les réseaux de Petri [13] et leurs extensions possèdent un intérêt fondamental en ayant fourni les premières approches de modélisation basées sur un support graphique facilitant l'expression et la compréhension des mécanismes de base pour des systèmes communicants et enfin, en n'étant pas lié à un langage particulier, ils assurent l'indépendance de la modélisation vis-à-vis des implémentations. Rapidement, les premiers réseaux de Petri se sont révélés être néanmoins un modèle trop limité pour les concepteurs d'applications informatiques ou plus généralement industrielles. Cette limitation est due principalement à trois facteurs : il est impossible de modéliser des comportements similaires au moyen d'une seule représentation condensée ; l'analyse du réseau n'est pas paramétrable ; les conditions et les conséquences de l'évolution d'un réseau, de nature purement quantitative sont inadéquates pour modéliser des conditions et des évolutions qualitatives. Plusieurs extensions ont donc été introduites soit plus compacts en terme

de description, c'est-à-dire des abréviations qui n'augmentent pas le pouvoir d'expression mais améliore juste la simplicité des modélisations ; soit plus puissant en termes de pouvoir d'expression, c'est-à-dire des extensions qui permettent de décrire des mécanismes et des fonctionnement qui ne pouvaient se faire avec le modèle de base [12]. Nous pouvons citer les réseaux de Petri à arcs inhibiteurs, permettant le test à zéro d'une place, les réseaux de Petri à reset arcs permettant la remise à zéro d'une place, les réseaux de Petri temporels pour prendre en compte les contraintes temporelles, etc.

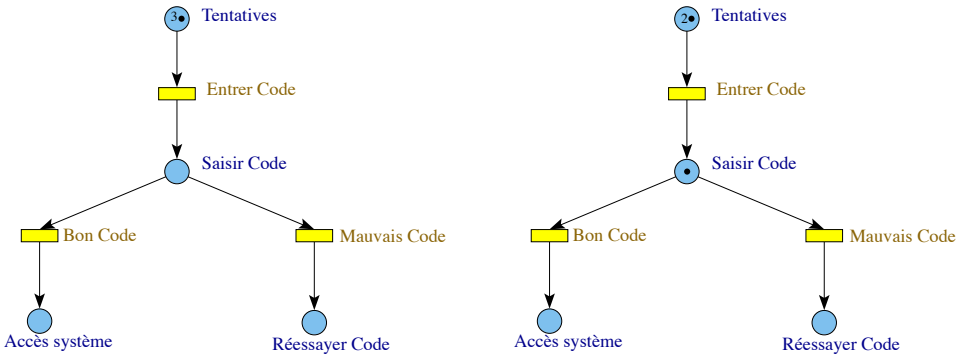
Pour conduire la vérification des systèmes, il est courant de construire le graphe d'états qui énumère de manière exhaustive les états possibles du système. Mais seulement, la construction de l'espace d'état (même fini) d'un système n'est toujours pas possible. L'une des raisons est l'explosion combinatoire des états due à la complexité du système, la forte concurrence dans le système, la présence d'un comportement infini dans le système, etc. Ce qui induit le problème de la limitation des ressources mémoires [1]. Les techniques dites à «ordre partiel» et plus particulièrement le dépliage [7] est une méthode d'ordre partiel largement utilisé. Il contourne le problème d'explosion combinatoire en éliminant la représentation du parallélisme par l'entrelacement des actions. Nous présentons dans ce papier une approche pour la construction du dépliage d'une extension particulière des réseaux de Petri : les réseaux de Petri à reset arcs.

Ce papier comporte six sections. La section 2 présente les réseaux de Petri en général et ceux à reset arcs en particulier, ce qui nous permet de définir dans la section 3 la notion de dépliage d'un réseau de Petri. Notre contribution commence dans la section 4 où nous présentons une approche pour construire le dépliage et le préfixe complet fini des réseaux de Petri avec des reset arcs et un algorithme pour identifier les processus de branchement d'un tel dépliage dans la section 5. La dernière section est consacrée à la conclusion et aux perspectives de ce travail.

2 Réseaux de Petri et les reset arcs

2.1 Réseaux de Petri

Un RdP [13] est un graphe biparti fait de deux types de sommets : les *places* et les *transitions*. Des arcs orientés relient des places à des transitions et *vice – versa* mais jamais deux sommets de même nature. Généralement, les places (ressources du système) sont représentées par des cercles qui sont constitués de jetons (instances des ressources), les transitions (événements du système) sont représentées par des rectangles ou des barres qui consomment et/ou produisent des jetons dans les places qui sont liées à la transition. Les places et les transitions sont reliés par des arcs orientés et valués (Place-Transition ou Transition-Place) qui indiquent le nombre de jetons à consommer et/ou à produire (par défaut le poids d'un arc est égal à 1). Le marquage d'un RdP est un vecteur à composantes entières positives ou nulles et dont la dimension est égale au nombre de places. La $n^{ième}$ composante de ce vecteur, représente le nombre de jetons dans la place n du RdP. La Figure 1(a) montre un exemple de réseau de Petri avec quatre places et trois transitions.



(a) Avant le tir de Entrer Code

(b) Après le tir de Entrer Code

FIGURE 1 – Exemple de réseau de Petri

Plus formellement,

Définition 1 (Réseau de Petri)**.** *Un réseau de Petri (normal) est un triplet* $N =< P,T,W >$ *avec :*

- P *un ensemble fini de places ;*
- T *un ensemble de transitions ;*
- $P \cap T = \emptyset$ *et* $P \cup T \neq \emptyset$
- $W : P \times T \cup T \times P \to \mathbb{N}$ *la fonction de valuation des arcs.*

Pour tout $(x,y) \in (P \times T)^2$, $W(x,y)$ désigne le poids de l'arc allant de x à y (s'il n'y a aucun arc, $W(x,y) = 0$). On définit aussi les fonctions *Pre* et *Post* respectivement les restrictions de W à $P \times T$ et à $T \times P$ ($Pre(p,t) = W(p,t)$ et $Post(p,t) = W(t,p)$). Pour tout $t \in T$, les pré-conditions de t, ${}^\bullet t = \{p \in P | W(p,t) > 0\}$ (reps. les post-conditions de t, $t^\bullet = \{p \in P | W(t,p) > 0\}$.

Un marquage M de N est l'application $M : P \to \mathbb{N}$ et le marquage initial est noté M_0 (le coupe $< N, M_0 >$ est appelé réseau marqué). Une transition $t \in T$ est sensibilisée par M, noté $M \xrightarrow{t}$, ssi $\forall p \in {}^\bullet t,\ M \geq W(p,t)$. Le tir de t conduit à un nouveau marquage M' ($M \xrightarrow{t} M'$) avec $\forall p \in P,\ M' = M - Pre(p,t) + Post(t,p)$. La Figure Figure 1(b) montre le réseau de Petri de la Figure 1(a) après le tir de la transition *Entrer Code*. Si le code entré est bon, que deviennent les jetons dans la place *Tentatives* ? On devait les retirer et les reset arcs nous permettent de réaliser le plus simplement cette action de retrait.

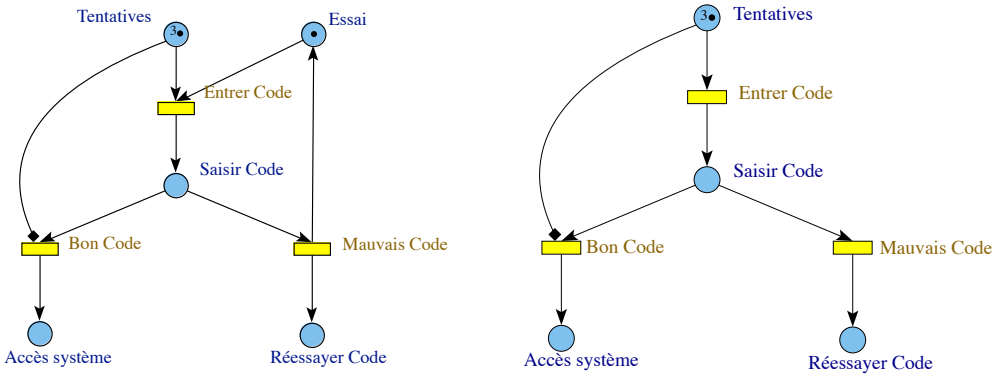
2.2 Réseaux de Petri avec des reset arcs

Les reset arcs constituent une des extensions des réseaux de Petri. Ils ne changent pas les règles de sensibilisation des transitions [5]. Par contre, si $M \xrightarrow{t} M'$ (i.e le tir de t fait passer le réseau de l'état M à l'état M') alors $\forall p \in P$ tel que $R(p,t) = 0$, $M'(p) = 0$. Mais si $W(t,p) > 0$ alors $M'(p) = W(t,p)$. De façon générale,

$$\forall p \in P,\ \ M' = (M - Pre(p,t))\ .\ R(p,t) + Post(p,t)$$

avec . le produit matriciel de Hadamard (produit terme à terme).
La Figure 2(a) représente un réseau de Petri avec un reset arc (l'arc reliant *Tentatives* à *BonCode*). C'est une modélisation d'accès à un système avec trois essais. Si le code est correct, on a accès au système sans avoir à retaper le code, sinon, dans la limite des trois tentatives, on est invité à retaper le code. Nous avons :

- P={Tentatives, Essai,SaisirCode, Accès Système ,RéessayerCode}, ;
- T={EntrerCode,Bon Code, MauvaisCode} ;
- R={(Essais, Bon Code)} ;
- M_0=(3,1,0,0,0).



(a) Version complète (b) Version simplifiée

FIGURE 2 – Réseau de Petri à reset arc

Le tir de la transition *EnterCode* conduit à un marquage $M_1 = (2,0,1,0,0)$ et ensuite le tir de *BonCode* conduit au marquage $M_2 = (0,0,0,1,0)$. Pour la suite et pour des raisons de simplification, nous utiliserons la version simplifiée de ce système tel représenté par la Figure 2(b).

Définition 2 (Réseau de Petri avec des reset arcs)**.** *Un réseau de Petri avec des reset arcs est un tuple* $N_R =< P,T,W,R >$ *avec* $< P,T,W >$ *un réseau de Petri tel défini dans la Définition 1 et* $R : P \times T \rightarrow \{0,1\}$ *est l'ensemble des reset arcs (*$R(p,t) = 0$ *s'il y a un reset arc qui relie p à t, sinon* $R(p,t) = 1$*).*

Lemme 1. *Les reset arcs n'enlève que des comportements au réseau de Petri.*

Preuve. Comme dit plus haut, les resets arcs n'influent pas sur la sensibilisation des transitions du réseau. Ainsi à la limite, le réseau de Petri avec des reset arcs comportement les mêmes comportement que le réseau sans reset arcs sous-jacent. De plus le tir de la transition intervenant dans un reset arc annule le contenu de la place. Ce qui pourrait empêcher la sensibilisation de certaines transitions qui aurait pu être sensibilisées dans le réseau sans reset arcs. Nous pouvons donc admettre le lemme 1. □

3 Dépliage des réseaux de Petri

Le dépliage des réseaux de Petri fait partir des méthodes dites d'ordre partiel qui permettent d'optimiser la vérification des systèmes finis. L'objectif du dépliage est de représenter explicitement l'ordre partiel des événements du système étudié par un réseau de Petri particulier. Cette catégorie de réseau de Petri correspond aux réseaux d'occurrence [2] et utilise une fonction d'homomorphisme [6].

3.1 Homomorphisme de réseaux

Définition 3 (Homomorphisme de réseaux)**.** *Soient* (N_1,M_{o1}) *et* (N_2,M_{o2}) *deux réseaux marqués avec* $N_i = \{P_i,T_i,W_i\}$ *pour* $i = 1,2$*. Soit h une application* $h : P_1 \cup T_1 \rightarrow P_2 \cup T_2$ *telle que* $h(P_1) \subseteq h(P_2)$, $h(T_1) \subseteq h(T_2)$*. h est un homéomorphisme de réseaux de* (N_1,M_{o1}) *vers* (N_2,M_{o2}) *si :*

- $\forall t_i \in T_1 : Pré_2(h(t_i)) = h(Pré_1(t_i))$;
- $\forall t_i \in T_1 : Post_2(h(t_i)) = h(Post_1(t_i))$;
- $M_{o2} = M_{o1}$.

L'homomorphisme de réseaux préserve donc les noeuds. Les deux premières conditions de la définition assurent que l'environnement des transitions est préservé par l'application h et la troisième impose à h que les marquages initiaux des deux réseaux soient les mêmes.

3.2 Réseau d'occurrence

Le dépliage des réseaux de Petri est représenté par une structure particulière de réseau de Petri.

Définition 4 (Réseau d'occurrence)**.** *Un réseau d'occurrence est un réseau* $O =< B,E,F >$ *tel que :*

- $|^\bullet b| \leq 1, \forall b \in B$;
- *O est acyclique ;*
- *O est fini par précédence, i.e* $\forall x \in B \cup E$, $\{ y \in B \cup E | y < x\}$ *est fini ;*
- *aucun élément n'est en conflit avec lui-même.*

Un réseau d'occurrence est un graphe sans circuit et chaque noeud est précédé par un nombre fini de noeuds. Trois types de relations, mutuellement exclusives, sont définis entre deux n$\frac{1}{2}$uds quelconques de O :

- la relation de causalité notée $\prec$: $\forall x,y \in B \cup E$ avec $x \neq y$, $x \prec y$ *ssi* le réseau contient au moins un chemin allant de x à y. Plus généralement, $x \preceq y$ *ssi* $x \prec y \vee x = y$;
- la relation de conflit notée $\perp$: $\forall x,y \in B \cup E$ avec $x \neq y$, $x \perp y$ *ssi* le réseau contient deux chemins $pt_1...x$ et $pt_2...y$ qui commence par la même place p et telle que $x \neq y$ $(x \perp y \Leftrightarrow y \perp x)$;
- et la relation de concurrence notée $\wr$: $\forall x,y \in B \cup E$ avec $x \neq y$, $x \wr y$ *ssi* $\neg((x \prec y) \vee (y \prec x) \vee (x \perp y))$ $(x \wr y \Leftrightarrow y \wr x)$.

Les comportements possibles d'un réseau d'occurrence sont capturés par la notion de configuration. La *configuration C* d'un réseau d'occurrence est un ensemble d'événements satisfaisant les deux conditions suivantes :

— $e \in C \Rightarrow \forall e' \preceq e,\ e' \in C$

— $\forall e, e' \in C,\ \neg(e \perp e')$

On désigne par $Conf(O)$ l'ensemble des configurations du réseau d'occurrence O et par $Cut(C)$ le marquage atteint par le franchissement des événements de C à partir du marquage initial : $Cut(C) = Min(N) + C^\bullet - {}^\bullet C$. Dans la pratique on considère les préfixes de dépliages et la notion de *configuration* est remplacée par celle de *configuration locale*. La configuration locale d'un événement e est notée $[e]$: $\forall e \in E, [e] = \{y \in B \cup E \mid y \prec e\}$.

3.3 Dépliage et préfixe du dépliage

Dans [6], les processus de branchement (ou processus arborescent) regroupent un ensemble de processus lié par une relation de causalité. Ils permettent de définir la notion de *dépliage* qui n'est rien d'autre que le plus grand processus de branchement (en général infini) qu'on peut construire pour un RDP. Le réseau résultant du dépliage est un réseau d'occurrence, un autre réseau de Petri où les places sont appelées *conditions* (étiquetés par les places correspondantes dans le réseau initial) et les transitions sont appelées *les évènements* (étiquetés par les transitions correspondantes dans le réseau initial). Pour un réseau marqué $< N, M_0 >$ et pour un réseau d'occurrence $O =< B, E, F >$, nous construisons le dépliage réseau marqué avec la procédure suivante.

1. Initialiser B avec les conditions initiales. Créer b_i (i allant de 1 à $|M_0|$) pour chacun des $|M_0|$ jetons et l'ajouter à B ;
2. Pour tout $t \in T$, s'il existe un $B' \subseteq B$ tel que $(B' =^\bullet t \wedge (\nexists e' \in E, {}^\bullet e' = B' \wedge (e') = t))$ alors :
   (a) Créer et ajouter l'événement e (lié à la transition t) à E ;
   (b) Créer et ajouter à B toutes les conditions liées aux post-conditions de t.
3. Répéter l'item 2 tant qu'il existe un événement e_n possible.

Le processus de construction peut donc s'exécuter indéfiniment si $< N, M_0 >$ permet un ou plusieurs comportements infinis. Alors même si le dépliage contourne le problème lié à la construction du graphe d'état [10], il n'en reste pas moins que c'est une structure potentiellement infinie alors qu'il est peu souhaitable d'avoir à manipuler de telle structure. Plus formellement,

Définition 5 (Dépliage)**.** *Le réseau marqué $< N, M_0 >$ (avec $N =< P, T, W >$) admet $Unf =< O, \lambda >$ comme dépliage ssi :*

- *$O =< B, E, F >$ est un réseau d'occurrence ;*
- *λ est une fonction d'étiquetage telle que $\lambda : B \cup E \to P \cup T$. Elle vérifie les propriétés suivantes :*
  — *$\lambda(B) \subseteq P$, $\lambda(E) \subseteq T$;*
  — *$\lambda(Min(O)) = m_0$;*
  — *Pour tout $e \in E$, la restriction de λ à ${}^\bullet e$ (resp. $e^\bullet$) est une bijection entre ${}^\bullet e$ (resp. $e^\bullet$) et ${}^\bullet\lambda(e)$ (resp. $\lambda(e)^\bullet$). Nous avons ${}^\bullet\lambda(e) = \lambda({}^\bullet e)$ et $\lambda(e)^\bullet = \lambda(e^\bullet)$, ce qui signifie que λ préserve l'environnement des transitions.*

La solution pour conduire des vérifications serait de ne considérer qu'un préfixe fini du dépliage du réseau qui contiendrait toutes les informations pertinentes. Plusieurs travaux [11, 8, 9, 7] ont permis de construire un tel préfixe fini complet des réseau de Petri en identifiant les événements (cut-offs) à partir desquels le dépliage n'est plus considéré. L'algorithme 1 permet de construire un tel préfixe. Il est prouvé [9] qu'un préfixe fini complet du dépliage existe toujours pour un réseau marqué même ceux présentant des comportements infinis. En pratique, la terminaison du calcul est basée sur la notion de *configuration locale* notée $[x]$. $[x] = \{y \in B \cup E \mid y \prec e\}$. La notion d'événement cut-off permet d'arrêter la dérivation de noeuds à partir d'un événement e donné du dépliage. Un événement cut-off est un événement déjà arriver dans la construction du dépliage et il est inutile de le représenter à nouveau.

Algorithme 1 : Algorithme du préfix fini et complet du dépliage

Entrées : Réseau de Petri marqué $< P,T,W,M_0 >$ avec $M_0 = \{p_1, p_2, ..., p_k\}$.
Sorties : Préfixe fini et complet du dépliage $Unf =< B,E,F >$ de $< N,M_0 >$ et les événements cut-offs co.
début
  $Unf := ((p_1,\emptyset),(p_2,\emptyset),...,(p_k,\emptyset))$;
  $pe := PE(Unf)$;
  $co := \emptyset$;
  tant que $pe \neq \emptyset$ **faire**
    Choisir un événement $e = (t,X)$ dans pe tel que [e] soit minimale;
    $pe := pe \backslash \{e\}$;
    si *e est cut-off* **alors**
      $co := co \cup \{e\}$;
    sinon
      Ajouter à Unf l'événement e et ses post-conditions de la forme (p,e);
      $pe := PE(Unf)$;
    fin
  fin
fin

4 Dépliage des réseaux de Petri avec des reset arcs

En prenant en compte le fait que la définition du dépliage des réseaux de Petri est bien formalisée dans la littérature, une première intuition serait de trouver un réseau de Petri normal, dont on construira le dépliage, équivalent au réseau de Petri à reset arcs. Mais à priori, trouver ce réseau équivalent est impossible. Si un tel réseau équivalent existait, on devait être capable, par exemple, de vérifier les mêmes propriétés au niveau des deux réseaux comme la *boundedness* ou l'accessibilité. Les deux propriétés sont indécidables pour un réseau de Petri à reset arcs, mais décidables pour un réseau de Petri normal [5]. Les réseaux de Petri à reset arcs sont plus puissants (réseau de haut niveau) et plus expressifs que les réseaux de Petri normaux. Pour notre part, pour pouvoir les déplier, il faudra alors raisonner à un haut niveau et nous le décrivons dans la Section 4.2. De ce point de vue, il est claire que la notion de réseau d'occurrence utilisée dans la Section 3.2 pour définir le dépliage des réseaux de Petri normaux ne nous permettra plus de définir celui avec des resets arcs. Nous introduisons dans ce papier la notion de R-réseau d'occurrence (R comme reset).

4.1 R-réseau d'occurrence

Un R-réseau d'occurrence est un réseau d'occurrence étendue aux reset arcs. C'est un réseau d'occurrence auquel on ajoute l'ensemble F_R formés de reset arcs. Plus particulièrement, un réseau d'occurrence est alors un R-réseau d'occurrence dont l'ensemble F_R est vide.

Définition 6 (R-réseau d'occurrence)**.** *Un R-réseau d'occurrence est un réseau $O_R =< O,F_R >$ tel que :*

— $O =< B,E,F >$ *est un réseau d'occurrence ;*
— $F_R : B \times E \rightarrow \{0,1\}$ *est un ensemble formé des reset arcs ($F_R(b,e) = 0$ s'il y a un reset arc qui relie b à e, sinon $F_R(b,e) = 1$).*

4.2 Dépliage des réseaux de Petri à reset arcs

La définition du dépliage d'un réseau de Petri avec des reset arcs est similaire à cette d'un réseau de Petri mais en remplaçant le réseau d'occurrence par le R-réseau d'occurrence. En nous basant sur le lemme 1, nous pouvons alors construire le dépliage d'un réseau de Petri à reset arcs de la manière suivante (Algorithme 2) :

Algorithme 2 : Algorithme du préfix fini et complet du dépliage d'un réseau de Petri avec reset arc

Entrées : Réseau de Petri avec reset arcs marqué $< P,T,W,R,M_0 >$ avec $M_0 = \{p_1, p_2, ..., p_k\}$.
début

1. Unf_R := Dépliage de $< N_R, M_0 >$ en ignorant les reset arcs;
2. $\forall (b,e) \in B \times E$, si $W(\lambda(b),\lambda(e)) = 0 \cap (b \notin [e] \cup e \notin [b])$ alors créer le reset arc (b,e) et l'ajouter à F_R i.e $F_R(b,e) = 0$;

fin

Plus précisément, le dépliage d'un réseau de Petri avec reset arcs ($N_R =< P,T,W,R >$) est un R-réseau d'occurrence ($O_R =< B,E,F,F_R >$) obtenu en dépliant le réseau de Petri avec reset arcs en ignorant les reset arcs (Algorithme 1). Puis pour toute condition b de B et pour tout événement e de E, si $R(\lambda(b),\lambda(e)) = 0$ alors $F_R(b,e) = 0$. Autrement dit, pour chaque place p et pour chaque transition t, relier par un reset arc, chaque copie de p et chaque copie de t dans le dépliage obtenu en ignorant les reset arcs. Mais si $e \in [b]$ ou $b \in [e]$, il n'est pas nécessaire de relier b et e par un reset arc (definition du dépliage).

Preuve. Le lemme 1 montre que les reset arcs n'enlèvent que des comportements au réseau de Petri. A l'item 1, on construit le dépliage du réseau de Petri sous-jacent. On a ainsi la garantie que tous les comportements du réseau de Petri à reset arcs y sont représentés. Avec l'item 2, on remet judicieusement les reset arcs pour enlever les comportement indésirables. On obtient bien avec l'algorithme2 le dépliage du réseau de Petri à reset arcs. Il en est de même pour le préfixe fini et complet du dépliage. On considère dans ce cas le préfixe fini et complet du réseau de Petri normal sous-jacent). □

La Figure 4.2 montre le dépliage du réseau de Petri à reset arcs de la Figure 2(b)

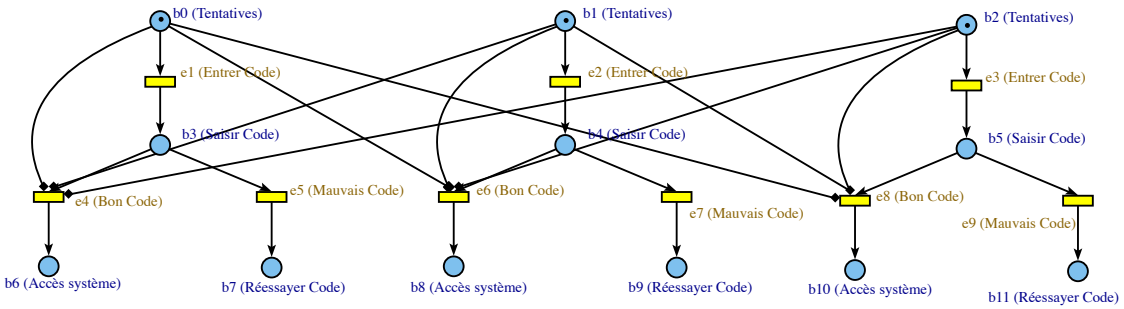


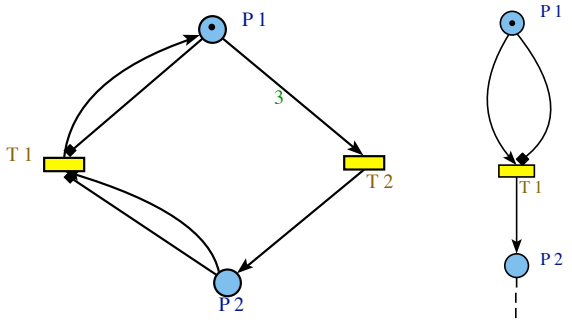
FIGURE 3 – Préfixe fini et complet du dépliage de réseau de la Figure 2(b)

4.3 Limites de l'approche

La première limite est liée au fait qu'on ne considère que les réseaux de Petri à reset arcs ayant des réseaux de Petri normaux sous-jacents bornés. Les reset arcs sont utilisés parfois pour borner les réseaux de Petri (Figure 4(a)) et *a priori* il n'y a pas de raison pour ne pas construire le dépliage mais en passant d'abord par le réseau normal sous-jacent, on ne peut plus construire ce le dépliage. La raison est que le réseau de Petri normal sous-jacent n'est pas borné.

La deuxième 'limite' est liée au fait qu'on ne peut pas garantir d'avoir le plus petit motif du dépliage. Prenons l'exemple du réseau de la Figure 4(b), en considérons le réseau de Petri normal sous-jacent, on commence la construction du dépliage par la représentation de trois conditions liées à $P1$ et donc la possibilité de produire trois événements liés $T1$. Mais seulement le comportement réel du réseau de Petri avec le reset arc, l'événement lié à

$T1$ ne peut qu'être réalisé qu'une seule fois. Le fait de remettre les reset arcs dans le dépliage obtenu permet de s'assurer de ces contraintes. Le motif obtenu n'est pas forcément pas le plus petit mais représente parfaitement le comportement du réseau. La deuxième limite n'en est vraiment pas une.



(a) (b)

FIGURE 4 – Exemples de limites

5 Calcul des processus de branchement

De part sa structure arborescente, le dépliage permet de vite calculer les processus (appelés processus de branchement) d'un réseau de Petri. Pour le dépliage d'un réseau de Petri normal, trois relations existent entre ses événements (causalité, concurrence et conflit). Pour un réseau de Petri normal, un processus de branchement est un ensemble d'événements liés par deux relations (causalité et concurrence), pas de conflit. Mais le dépliage d'un réseau de Petri à rest arcs, on observe un autre comportement, que l'on matérialisera par la définition d'une nouvelle relation (relation de reset) qui pourrait empêcher deux événements, même n'étant pas en conflit, de ne pas appartenir à un même processus. C'est le cas pas exemple, dans l'exemple de la Figure 4.2 des événements $e4$ et $e2$. La présente de $e4$ pourrait inhiber l'événement $e2$.

Définition 7 (Relation de reset)**.** *e_1 et e_2 sont dans une relation de reset ($e_1 \top e_2$) ssi $\exists\, b \in {}^\bullet e_2$ tel que $(b, e_1) \in F_R$. De façon générale, $e_1 \top e_2$ ssi $\exists\, e \in E$ tel que $((e \prec e_2) \wedge (e_1 \top e))$. $\top$ n'est pas commutative ; Si $(e_2 \prec e_3)$ et $(e_1 \top e_2)$ alors $(e_1 \top e_3)$.*

Nous pouvons alors formaliser la notion de processus de branchement pour un réseau de Petri à reset arcs comme étant un ensemble d'événements en relations de causalité et de concurrence et dans lequel il ne subsiste aucun conflit et pour lequel deux événements ne sont en relation de reset. Soit C_i un ensemble formé par les conditions et les événements du dépliage et construit comme suit :

1. $Min(O) \subseteq C_i$
2. Ajouter un $e_i \in E$ à C_i ssi :
   — $e_i \notin C_i$;
   — $\exists b \in C_i$ *tel que* $e_i \in b^\bullet$;
   — $\forall e_j \in E, \neg(e_j \perp e_i) \vee \neg(e_j \top e_i)$.
3. Si e_i est ajouté à C_i alors $C_i = C_i \cup e_i^\bullet$
4. Répétez l'étape 2 autant de fois.

L'ensemble des événements $E_i \subseteq E$ de C_i définit un processus de branchement. Pour l'exemple de la Figure 4.2, nous pouvons identifier les processus suivants :
— $P_1 = \{e_1, e_4\}$
— $P_2 = \{e_1, e_5, e_4, e_2, e_6\}$

— $P_3 = \{e_1, e_5, e_4, e_2, e_7, e_3, e_8\}$
— $P_4 = \{e_1, e_5, e_4, e_2, e_7, e_3, e_9\}$

6 Conclusion et perspectives

Dans ce travail, nous avons présenté une approche pour construire le dépliage des réseaux de Petri à reset arcs et un algorithme pour construire son préfixe complet et fini. Cet algorithme permet d'étendre alors la notion de dépliage à une extension particulière des réseaux de Petri, les réseaux de Petri à reset arcs. L'intérêt de cette approche est de pouvoir construire le dépliage des réseaux de Petri avec des reset arcs sans avoir à trouver une quelconque équivalence avec un réseau de Petri.

Nos travaux visent à définir une algèbre des processus de branchement [3, 4], les processus issus du dépliage des réseaux de Petri. La perspective de ce travail est de pouvoir exprimer le dépliage des réseaux de Petri à reset arcs sous forme algébrique, d'en déduire les processus (voire les processus maximaux) et de trouver des équivalences de conflits.

Références

[1] Bernard Berthomieu and Francois Vernadat. State class constructions for branching analysis of time petri nets. In *In TACAS ?2003, volume 2619 of LNCS*, pages 442–457. Springer Verlag, 2003.

[2] Thomas Chatain and Claude Jard. Complete finite prefixes of symbolic unfoldings of safe time petri nets. In Susanna Donatelli and P.S. Thiagarajan, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2006*, volume 4024 of *Lecture Notes in Computer Science*, pages 125–145. Springer Berlin Heidelberg, 2006.

[3] Delfieu D., M. Comlan, and Sogbohossou. Algebraic analysis of branching processes. In *Sixth International Conference on Advances in System Testing and Validation Lifecycle*, pages 21–27, 2014. Best paper award.

[4] Delfieu D. and M. Sogbohossou. An algebra for branching processes. In *(CoDIT, 2013 International Conference on*, pages 625–634, May 2013.

[5] Catherine Dufourd, Petr Jančar, and Philippe Schnoebelen. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *Proceedings of the 26th ICALP'99*, volume 1644 of *Lecture Notes in Computer Science*, pages 301–310, Prague, Czech Republic, July 1999. Springer.

[6] Joost Engelfriet. Branching processes of petri nets. *Acta Informatica*, 28(6) :575–591, 1991.

[7] Javier Esparza and Keijo Heljanko. Unfoldings - a partial-order approach to model checking. *EATCS Monographs in Theoretical Computer Science*, 2008.

[8] Javier Esparza, Stefan Römer, and Walter Vogler. *An Improvement of McMillan's Unfolding Algorithm*. Mit Press, 1996.

[9] Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3) :285–310, 2002.

[10] Kenneth L McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Computer Aided Verification*, pages 164–177. Springer, 1993.

[11] K.L. McMillan and D.K. Probst. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1) :45–65, 1995.

[12] Diaz Michel. *Vérification et mise en oeuvre des réseaux de Petri*. Traité IC2. Hermès Science, Paris, 2003.

[13] Carl Adam Petri. Communication with automata. 1962.